15/03/2801 #3

Europäisches Patentamt | European Patent Office | Office européen des brevets

RECD PCT/PTO 13 JAN 2005

Bescheinigung | Certificate | Attestation

REC'D 02 OCT 2003
WIPO PCT

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. Patent application No. Demande de brevet n°

02291778.5

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH
RULE 17.1(a) OR (b)

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

R C van Dijk

BEST AVAILABLE COPY

Anmeldung Nr:
Application no.: 02291778.5
Demande no:

Anmeldetag:
Date of filing: 15.07.02
Date de dépôt:

Anmelder/Applicant(s)/Demandeur(s):

SCHLUMBERGER Systèmes
50, avenue Jean Jaurès
92120 Montrouge
FRANCE

Bezeichnung der Erfindung/Title of the invention/Titre de l'invention:
(Falls die Bezeichnung der Erfindung nicht angegeben ist, siehe Beschreibung.
If no title is shown please refer to the description.
Si aucun titre n'est indiqué se referer à la description.)

USB device

In Anspruch genommene Priorität(en) / Priority(ies) claimed /Priorité(s) revendiquée(s)
Staat/Tag/Aktenzeichen/State/Date/File no./Pays/Date/Numéro de dépôt:

Internationale Patentklassifikation/International Patent Classification/
Classification internationale des brevets:

G06F13/00

Am Anmeldetag benannte Vertragstaaten/Contracting states designated at date of filing/Etats contractants désignées lors du dépôt:

AT BE BG CH CY CZ DE DK EE ES FI FR GB GR IE IT LI LU MC NL PT SE SK TR

USB device

FIELD OF THE INVENTION

The invention relates to a system comprising a first device arranged to communicate with a second device. The first device can be, for example, a USB host. The second device can be, for example, a USB device, which communicates with the USB host via a USB bus using the USB protocol.

BACKGROUND OF THE INVENTION

The system comprises a USB host, which is connected to various USB devices via a USB bus. The USB host communicates with the USB devices with the USB protocol. The USB protocol allows connecting several USB devices on the same USB bus using a system of time-sharing based on addressed devices. The USB protocol is organized as a master/slave architecture, the USB host is thus responsible of the time-sharing management.

The USB host may comprise various applications. One or several services may be needed to run an application. An application uses one or several drivers to access and use the associated services. The drivers may be on the USB host.

The USB device may comprise various services, in particular those, which are needed by the applications of the USB host. A service may be offered, for example, at the device level (standard USB device), or at the interface level (composite USB device).

The USB device is organized into several levels a device level, a configuration level, an interface level and an endpoint level. Each level is represented by different USB descriptors:

- A device descriptor describing the overall device. The device descriptor may be associated to one or more configuration descriptors.
- A configuration descriptor describing the electrical characteristics of the USB device, or of a part of the USB device. The configuration descriptor may be associated to one or more interface descriptors.
- An interface descriptor describing a particular service of the USB device. An interface may contain one or more alternate settings. The interface descriptor may be associated to zero or more endpoint descriptors.
- An endpoint descriptor describing a communication channel used by the service defined by the interface descriptor.

In a plugging step, the USB device is plugged onto a USB port of the USB host.

In an enumeration step, all the USB descriptors are then retrieved from the USB device to the host device. The enumeration step is triggered off with the modification of the voltage level on the line D+ or D- (depending on the USB device speed) due to a pull-up resistor present in the USB device on one of the lines.

In a loading step, the USB Host then uses the descriptors to load all the drivers of the USB device. The number of drivers loaded depends on the number of different services present in the USB device. For example, if the USB device is at the same time a scanner

and a printer the USB device will have to present two interfaces during the enumeration step. In that case, two drivers, one associated to the scanner interface, and the other one associated to the printer interface, will be loaded. A main driver associated to the device itself could also be loaded.

SUMMARY OF THE INVENTION

An object of the invention is to reduce the costs.

According to one aspect of the invention, a method of configuring a system comprising a main device and an auxiliary device arranged to co-operate with each other, the main device being arranged to handle one or more functionalities, the auxiliary device being arranged to effect one or more functionalities, characterised in that the method comprises an adaptation step, in which the auxiliary device is made to hide from the main device at least those of its functionalities that the main device cannot handle.

The first device can be, for example, a USB host. The second device can be, for example, a USB device. The functionalities that the main device cannot handle will be hidden from the main device. Thus the invention allows a reduction of the costs.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 illustrates a system comprising a USB host and a USB device.

Fig.2 illustrates the structure of the USB device;

Fig. 3 illustrates a method of using the system; and

Fig. 4 illustrates a system comprising a USB host and a USB device;

Fig. 5 illustrates a method of using the system;

Fig. 6 illustrates a system according to the invention;

Fig. 7 illustrates a method according to the invention; and

Fig. 8 illustrates a method according to the invention.

DETAILED DESCRIPTION

As illustrated in figure 1, the invention will be explained in the context of a system using the USB protocol. The system comprises a USB host, which is connected to a USB device via a USB bus. The USB host communicates with the USB devices with the USB protocol.

The communication between the USB host and the USB device can be made according four different transfer modes:

- a control transfer, offering mainly a delivery and data integrity guarantee,
- an interrupt transfer, offering mainly a periodicity and data integrity guarantee,
- a bulk transfer, offering mainly a data integrity guarantee, and a possibly good data rate,
- a isochronous transfer, offering mainly a bandwidth guarantee.

Two of these four modes require a bandwidth reservation, which is accorded or not by the USB host

after an enumeration phase, depending on the bandwidth already reserved by other USB devices, which are plugged onto the USB bus.

The USB host comprises various applications (A1, A2). One or several services (S1, S2) are needed to run an application (A1). The application A1 needs, for example, to use the service S1 and the service S2. The services (S1, S2, S3, S4) are located on the USB device. To use a specific service S1, an application A1 may use a driver D1. The USB host also comprises a standard application (A0) associated with a standard driver (D0). Advantageously the standard application (A0) is implemented on a big number of USB hosts.

The USB device comprises various services (S1, S2, S3 and S4) in particular those (S1, S2, S3), which are needed by the applications (A1, A2) of the USB host. As illustrated in figure 2, a service (S) can be offered, for example, at the device level, or at the interface level (S1,S2,S3,S4). The USB device also comprises a standard service (S0). Advantageously the standard service (S0) is implemented on a big number of USB devices.

As illustrated in figure 3, in a connecting step CON, the USB device is connected to a USB port of a USB host.

In a first checking step CHECK1, the USB device checks whether a negotiation flag is activated or not.

If not:

- in a first enumerating step ENUM1, the USB host will enumerate the USB device. In other words, as illustrated in figure 2, the USB host will retrieve from the USB device to the USB host only the descriptors (I) associated

to the standard service S0,

- in a loading step LOAD, the standard driver D0 is loaded into an active memory of the host,
- in a negotiation step NEGO, the standard application A0 negotiates the services (S1, S2, S3) to activate. The negotiating step comprises the following sub-steps:
  - o a receiving step, in which the standard application A0 receives from the standard service S0 a first list of all the different services (S1, S2, S3, S4) which are available on the USB device,
  - o a comparing step, in which the standard application compares the first list of all the different services (S1, S2, S3, S4) which are available on the USB device with a second list of the services (S1, S2, S3) needed by the applications (A1,A2) of the USB host to deduce the services to be activated (S1, S2, S3) on the USB device,
  - o a service activating step, in which the USB device activates the services to be activated, for example, by disconnecting and reconnecting the USB device to the USB host.

- In an flag activating step ACTIV, the negotiation flag is activated.
- In an initialization step INIT, the USB device removes its pull-up resistor in order to detach itself and then re-attach itself.

In a second checking step CHECK2, the USB device checks whether the negotiation flag is activated or not.

If yes:

- in a deactivating step DEACTIV, the negotiation flag is deactivated,
- in a second enumerating step ENUM2, the USB host enumerates the USB device. As illustrated in figure 2, only the descriptors (II) associated to the services (S1, S2, S3) which have been activated and the descriptor associated to the standard service (S0) will be retrieved,
- in a second loading step LOAD2, the standard driver D0 and the drivers associated to the services (S1, S2, S3) which have been activated are loaded into the active memory of the USB host.

The USB device is now ready for use.

According to an advantage of the invention, if a new service has to be added on the USB device, the standard service (S0) does not change and therefore the standard application (A0) does not change. The invention thus allows a reduction of the cost.

As illustrated in figure 4 and 5, if the USB device is already plugged, and the user starts a new application (A3), which requires a new service (S4), which is not activated in the USB device, the standard application (A0) can negotiate the activation of the new service (S4) in a new negotiating step.

In an opening step OPEN, user opens a new application (A3) requiring a service (S4), which is not available in the current configuration of the USB device.

In a negotiating step NEGO, the USB host

activates the service (S4).

In a flag activating step ACTIV, the USB device activates the negotiation flag.

In an initialization step INIT, the USB device removes its pull-up resistor in order to detach itself and then re-attach.

In a checking step CHECK, the USB device checks whether the negotiation flag is activated or not.

If yes:

- in a deactivating step DEACTIV, the negotiation flag is deactivated,
- in an enumerating step ENUM, the USB host enumerates the USB device. As illustrated in figure 2, only the descriptors (III) associated to the services (S1, S2, S3, S4) which have been activated and the descriptor associated to the standard service (S0) will be retrieved,
- in a loading step LOAD, the standard driver D0 and the drivers associated to the services (S1, S2, S3,S4) which have been activated are loaded into the active memory of the USB host.

The device is ready for use.

The description hereinbefore illustrates the following features:

The invention concerns a method of configuring a system. The system comprises a main device and an auxiliary device. The main device and the auxiliary device are arranged to co-operate with each other. The main device is arranged to handle one or more functionalities. The auxiliary device is arranged to

effect one or more functionalities. The method is characterised in that the method comprises an adaptation step, in which the auxiliary device is made to hide from the main device at least those of its functionalities that the main device cannot handle.

The first device can be, for example, a USB host in particular a computer, a PDA or GSM.

The second device can be, for example, a USB device in particular a smart card or more generally any device that can be personalized, for example, a PDA or GSM.

The USB device can be, for example, a Smart Card comprising three different services:

- Keys and rights management (APDU command transport) as service [S0],
- Document signature as service [S1],
- Data streaming application (DRM) as service [S2].

The Smart Card can be used in different USB hosts non-exhaustively listed hereafter:

- Corporate Personal Computers running Windows XP as environment [E1],
- Home Personal Computers running Windows XP as environment [E2],
- GSM (also as USB On-The-Go device) as environment [E3],
- PDA (as USB On-The-Go device) as environment [E4].

For all these USB hosts, the services that can be accessed could be:

[S0] and [S1] for [E1], because the user is not administrator of the machine, and can not install a new

driver,

[S0], [S1] and [S2] for [E2], because the user is administrator of the machine and can install any service available,

[S0] only for [E3], for memory or consumption economy reasons,

[S0] only for [E4], because the host cannot be personalized and only the driver for [S1] is available.

The services located on the USB device are not necessarily related to the USB device itself. The services can be vendor specific or not. For example, a smartcard (USB device) may have a modem service, so that the smartcard can be seen as a distant terminal through a modem connection.

Figure 6 illustrates another example wherein the smartcard (USB device) may also comprise a loudspeaker service, a microphone service and a decryption service. Once the smartcard (USB device) is plugged into a computer (USB host) the loudspeaker service will be activated by the computer. The computer will first consider the smartcard (USB device) as a loudspeaker and will send an encrypted music file (1). The loudspeaker service will receive the music file and send it (2) to the decryption service for decrypting the music file. Then the decrypted music file is sent (2) to the microphone service so that the computer (USB host) believes now (3) that the smartcard (USB device) is a microphone wherein someone is speaking. The computer will then send the decrypted music file to the real loudspeaker (4).

It should be clear that the invention is not limited to devices communicating using the USB protocol. Other protocol like, for example, firewire based protocol may be used.

It should be clear that the invention is not

limited to devices communicating according to a master/salve protocol.

In the comparing step, the first list of services and the second list of services have been used to deduce the services to be activated on the USB device. More generally any other set of data defining the services can be used to deduce the services to be activated. It can be, for example, a set of data identifying various services and giving the bandwidth needed by these services.

These and other aspects of the invention are described in the following section entitled "memo".

MEMO

The USB protocol allows to connect several devices on the same bus using a system of time sharing based on addressed devices. Since the protocol is organized as a pure master/slave architecture, the USB host is responsible of the time sharing management.

A USB device is organized with several levels, each level being represented by a USB descriptor :

- the *device descriptor* describes the overall device. It is associated to one or more configuration descriptors,
- the *configuration descriptor* describes some electrical characteristics of the device, or a part of the device. It is associated to one or more interface descriptors,
- the *interface descriptor* describes a particular application of the device. An interface can contain one or more alternate settings. An interface descriptor is associated to zero or more endpoint descriptors,
- the *endpoint descriptor* describes a communication channel used by the application defined by the interface descriptor.

The USB device is seen by the USB host as a services provider. This service(s) can be offered at the device level (standard USB device), or at the interface level (composite USB device).

The USB Host uses the descriptors to load one or more driver(s) for the device, and configure the associated applications. The descriptors are retrieved when the device is plugged onto the USB bus, during an "enumeration phase". This enumeration

phase is triggered off with the modification of the voltage level on the line D+ or D- (depending on the USB device speed) due to a pull-up resistor present in the device on one of the lines.

The number of drivers loaded depends on the number of different services present in the device. For example, if a device is at the same time a scanner and a printer, from a USB point of view, it will have to present two interfaces during the enumeration phase. In that case, two drivers, one associated to the scanner interface, and the other one associated to the printer interface, will be loaded after the enumeration phase. A third driver associated to the device itself can also be loaded (composite device driver relative to the Manufacturer and Product ID of the device).

The endpoints can use one of the four following transfer modes :

- the control transfer, offering mainly a delivery and data integrity guarantee,
- the interrupt transfer, offering mainly a periodicity and data integrity guarantee.
- the bulk transfer, offering mainly a data integrity guarantee, and a possibly good data rate,
- the isochronous transfer, offering mainly a bandwidth guarantee.

Two of these four modes require a bandwidth reservation which is accorded or not by the USB host after the enumeration phase, depending on the bandwidth already reserved by the other devices plugged onto the bus.

Autoadaptative USB device :

Following the scheme described above, some points could be improved with an "auto-adaptative" device :

- All the necessary drivers must be installed in the USB host in order to be able to run correctly the device. When a driver corresponding to a device interface is not found, the USB host asks the user to install this driver from an external source (floppy disk, network ...). If the driver can not be installed as it is the case if the driver is not available, or the host is not configurable (Windows 2000/XP environments when the user is not the administrator of the machine, USB-On-The-Go devices), the device is considered as not correctly installed,
- Even in the case where the drivers are present in the USB host, they are loaded as soon as the device is plugged, even if they are not used,
- Depending on its structure (in term of interfaces, endpoints ...), the device attachement can be refused by the host, due to a bandwidth lack,
- It is not possible to add or modify easily a communication interface for a device once the driver is defined and installed.

The idea is to define a device able to change its configuration on a host request.

To do so, the device should present a default interface when plugged onto a USB port. This default interface should be able to manage a "negocitation" between the host and the device in order to define the services that should be activated for the current session.

The following diagram shows a possible negotiation phase : see figure 7

Another possible case is the modification of the device configuration during a working session.
For example, if the device is already plugged, and the user starts an application wich requires a particular service not activated in the device, the default device driver needs to negotiate a new configuration for the device : see figure 8

Another advantage, not listed above, is the USB-IF (USB Impementer Forum) defines USB Device Classes. All the USB Host must contain default drivers for the defined Classes. When a Device Class is modified because some new functionalities are added for example, it is difficult to update all the USB hosts.
The method described above allows to add some functionalities to a device without modifying the Device Class, and so, without modifying the standard drivers of the USB hosts.

Example :

The device is a Smart Card embeding three different services :
- Keys and rights management (APDU command transport) as service [S1],
- Document signature as service [S2],
- Data streaming application (DRM) as service [S3].

This Smart Card can be used in different environment non exhaustively listed hereafter :
- Corporate Personal Computers running Windows XP as environment [E1],
- Home Personal Computers running Windows XP as environment [E2],
- GSM (also as USB On-The-Go device) as environment [E3],
- PDA (as USB On-The-Go device) as environment [E4].

For all these platforms, the services that can be accessed are different :
- The company, owner of [E1], only installs the drivers for [S1] and [S2],
- The Smart Card holder, owner of [E2], installs all the drivers associated to [S1], [S2] and [S3],
- The GSM provider, owner of [E3], installs [S1] and [S3],
- The PDA provider, owner of [E4], installs only [S1].

When the Smart Card is plugged onto a USB port (on any environment), it presents only the interface associated [S1] considered as default interface.
During the enumeration phase, the USB host loads the driver associated to [S1] (default or main driver), and the driver starts the negotiation phase.

Several phases are necessary in the negotiation phase :
- *Actors identification* : a USB request allows the Smart Card to indicate what are its services, and there state (activated or not). The Smart Card can also indicate if the negotiation phase has already been performed. This can be done for example using a Class or Vendor Specific Descriptor,
- *Service choice* : the USB host choses the services that will be available during the working session, taking care of the drivers available and of the device nature, and informs the Smart Card using a USB request,
- *Configuration change* : The USB Host validates the negotiation phase using a USB request, and the Smart Card detaches itself by removing the pull-up resistor from the D+ or D- line.

After the negotiation phase, the Smart Card appears as "negotiated" during the *actors identification* phase, what prevents the host to start the *service choice* phase.

For the environment described above, the services could be :
- [S1] and [S2] for [E1], because the user is not administrator of the machine, and can not install a new driver,
- [S1], [S2] and [S3] for [E2], because the user is administrator of the machine and can install any service available,
- [S1] only for [E3], for memory or consumption economy reasons,

- [S1] only for [E4], because the host can not be personalized and only the driver for [S1] is available.

For [E1], [S3] could be activated only if the DRM application is started by the user. In this case, the default driver starts a new *service choice* phase (disabling [S1] and activating [S3]), followed by a *configuration change* phase.
After this step, the service active for [E3] is [S3].

Remarks :

- This memo is applicable to any USB device,
- The memo describes a USB device that can be easily personalized with no modification of the default driver,
- The memo describes a process allowing to chose the services offered by a USB device without modifying the device itself.

CLAIMS

1. A method of configuring a system comprising a main device and an auxiliary device arranged to co-operate with each other, the main device being arranged to handle one or more functionalities, the auxiliary device being arranged to effect one or more functionalities, characterised in that the method comprises an adaptation step, in which the auxiliary device is made to hide from the main device at least those of its functionalities that the main device cannot handle.

2. The method according to claim 1, characterized in that the adaptation step comprises the following sub-steps:

- a notification step, in which the auxiliary device notifies the main device a set of data identifying the one or more functionalities that the auxiliary device can effect;
- a identification step, in which the set of data is used to identify the functionalities that the auxiliary device can effect but that the main device cannot handle; and
- a configuration step, in which the auxiliary device is configured to hide from the main device at least those of its functionalities that the main device cannot handle.

3. The method according to claim 2, wherein the adaptation step is followed by an enumeration step, in which the auxiliary device presents itself to the main

device without the functionalities identified in the identification step.

4. The method according to claim 1, wherein the adaptation step is carried out automatically when connecting the auxiliary device to the main device.

5. The method according to claim 3, wherein a simulation step is carried out between the adaptation step and the enumeration step, in which the disconnecting and the reconnecting of the auxiliary device is simulated.

6. The method according to claim 1, characterized in that the main device is a USB host and in that the auxiliary device is a USB device.

7. The method according to claim 1, characterized in that the auxiliary device is a smartcard.

SUMMARY

The invention concerns a method of configuring a system. The system comprises a main device and an auxiliary device. The main device and the auxiliary device are arranged to co-operate with each other. The main device is arranged to handle one or more functionalities. The auxiliary device is arranged to effect one or more functionalities. The method is characterised in that the method comprises an adaptation step, in which the auxiliary device is made to hide from the main device at least those of its functionalities that the main device cannot handle.

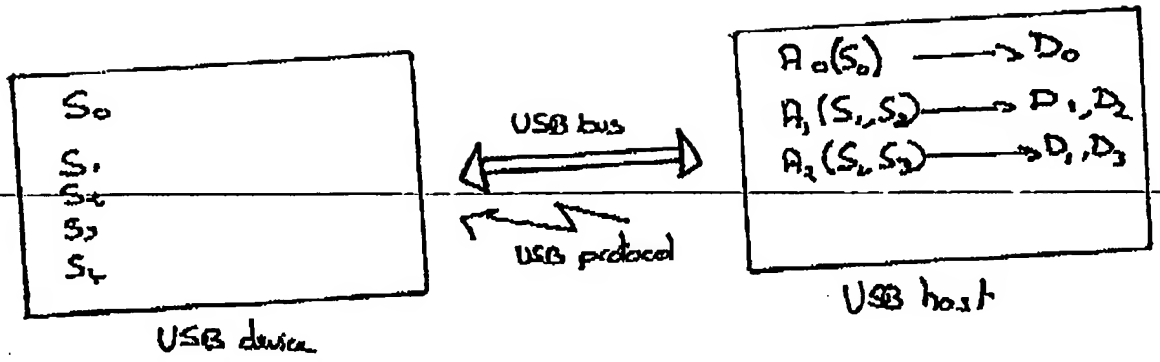
S_0
S_1
S_2
S_3
S_4
USB device
USB bus
USB protocol
$A_0(S_0) \longrightarrow D_0$
$A_1(S_1, S_3) \longrightarrow D_1, D_2$
$A_2(S_2, S_4) \longrightarrow D_1, D_3$
USB host

Fig 1

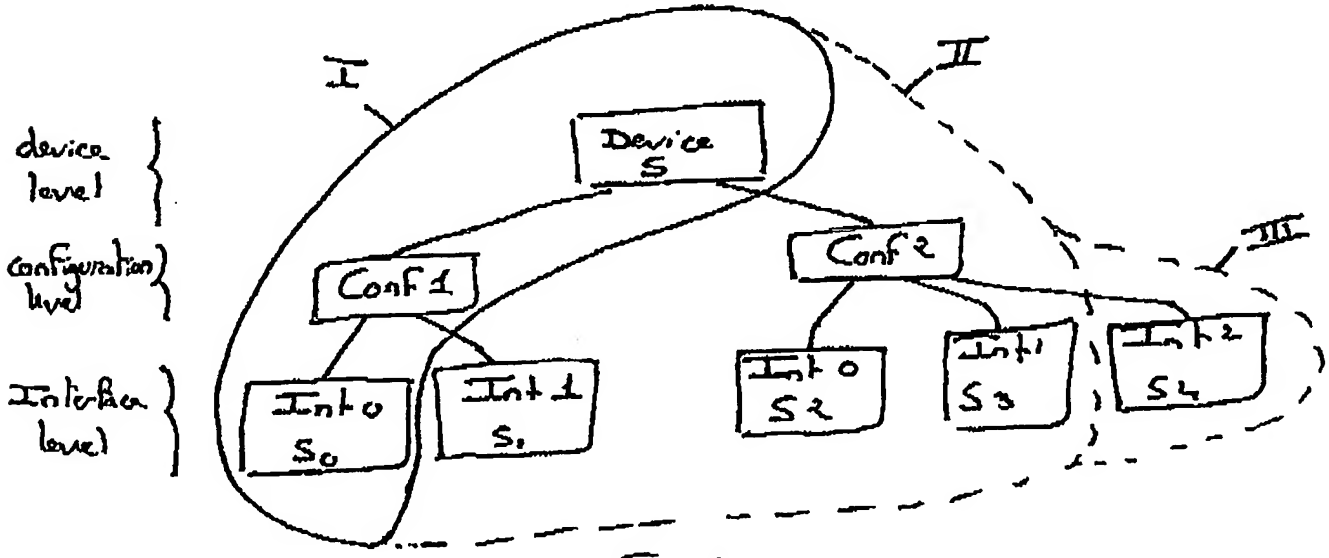
I
II
III
device level
configuration level
interface level
Device S
Conf 1
Conf 2
Int 0 S_0
Int 1 S_1
Int 0 S_2
Int 1 S_3
Int 2 S_4

Fig 2

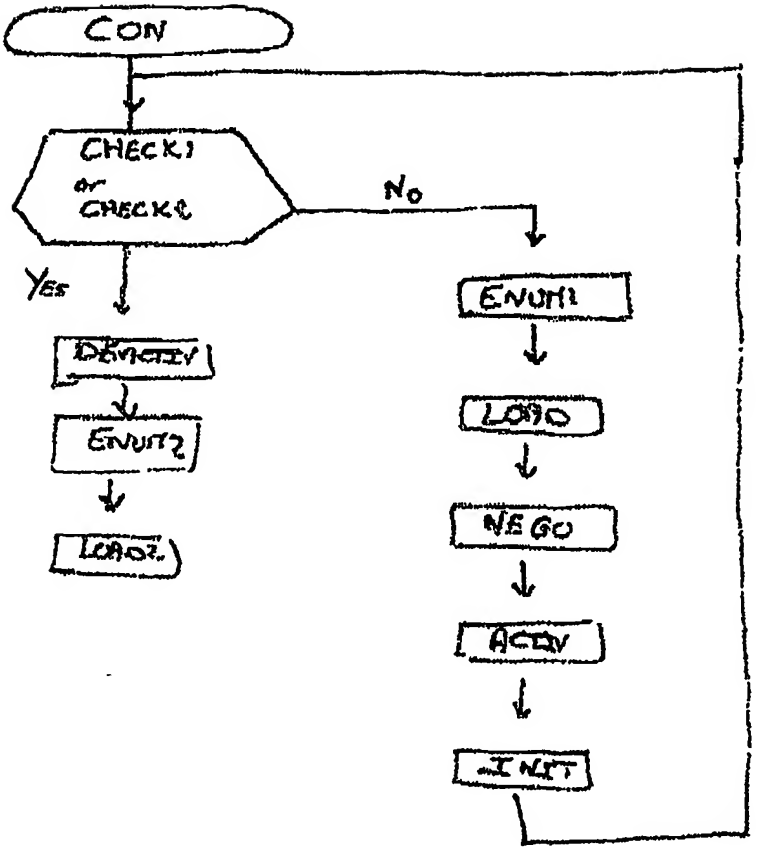
CON
CHECK1
or
CHECK2
No
Yes
ENUM1
LOAD1
NEGO
ACCN
INIT
DEVICE1
ENUM2
LOAD2

Fig 3

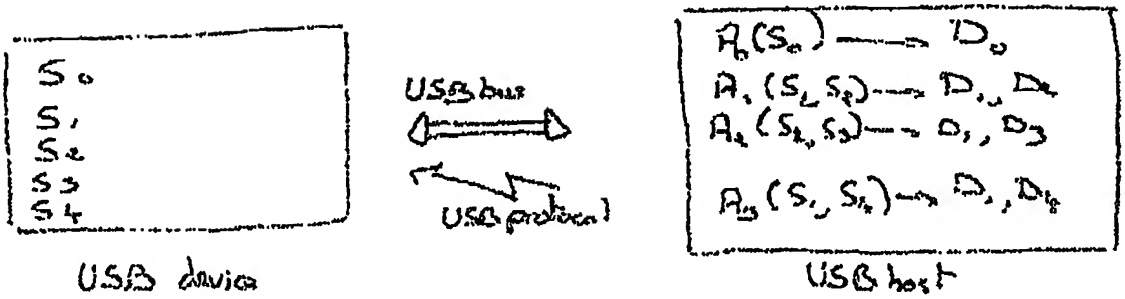
S0
S1
S2
S3
S4
USB bus
USB protocol
A0(S0) ——→ D0
A1(S1,S2) ——→ D1, D2
A2(S3,S2) ——→ D1, D3
A15(S1,S4) ——→ D1, D4
USB device
USB host

Fig 4

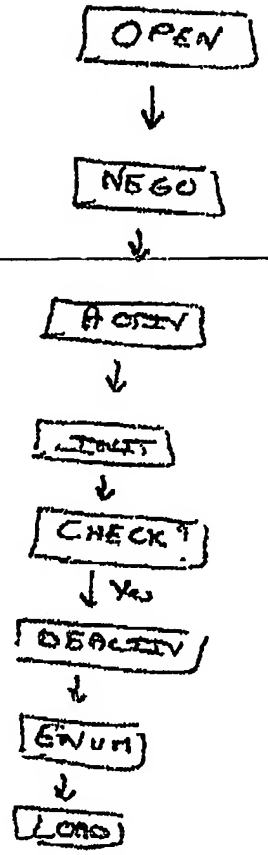

Fig 5

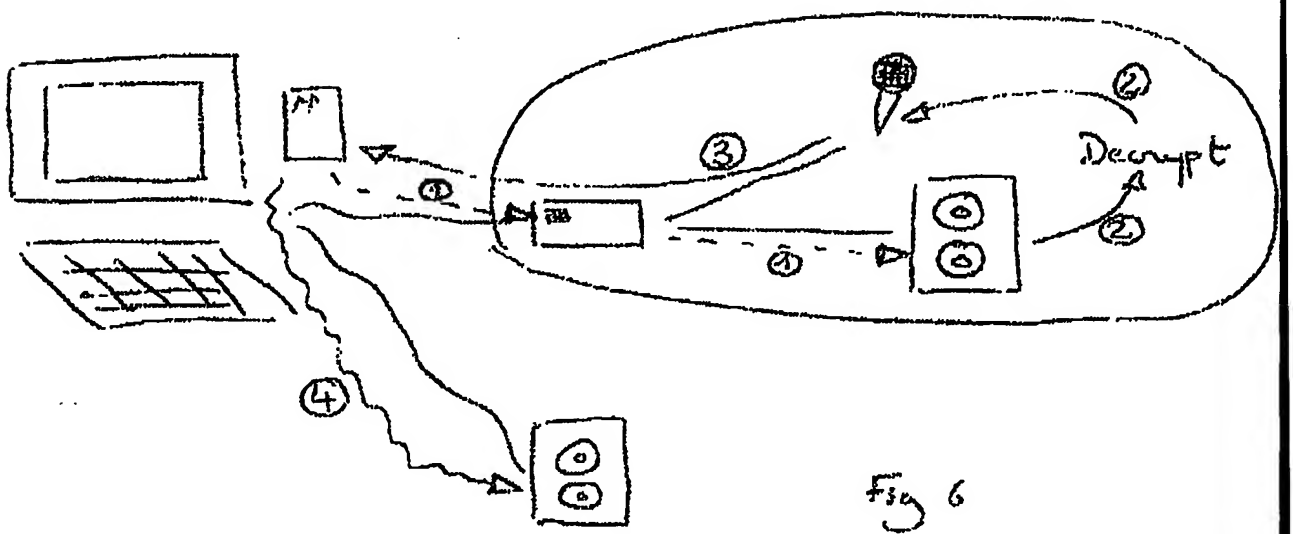

Fig 6

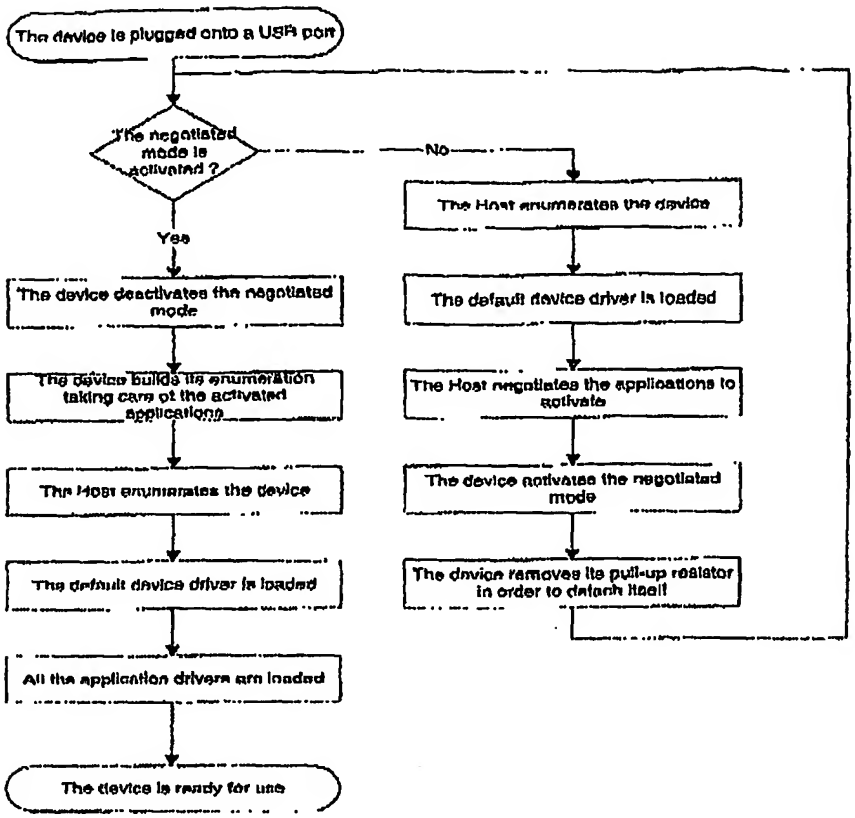
The device is plugged onto a USB port
The negotiated mode is activated ?
No
Yes
The Host enumerates the device
The default device driver is loaded
The Host negotiates the applications to activate
The device activates the negotiated mode
The device removes its pull-up resistor in order to detach itself
The device deactivates the negotiated mode
The device builds its enumeration taking care of the activated applications
The Host enumerates the device
The default device driver is loaded
All the application drivers are loaded
The device is ready for use

Fig 7

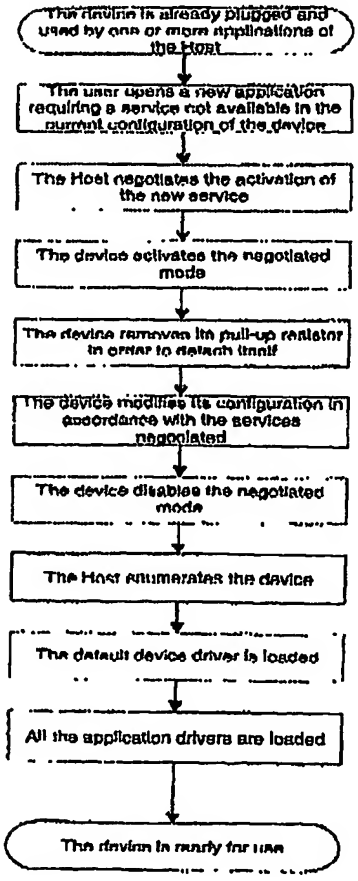
The device is already plugged and used by one or more applications of the Host
The user opens a new application requiring a service not available in the current configuration of the device
The Host negotiates the activation of the new service
The device activates the negotiated mode
The device removes its pull-up resistor in order to detach itself
The device modifies its configuration in accordance with the services negotiated
The device disables the negotiated mode
The Host enumerates the device
The default device driver is loaded
All the application drivers are loaded
The device is ready for use

Fig 8